THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 9, November 1982
Printed in U.S.A.

A 9.6-kb/s DSP Speech Coder

By R. E. CROCHIERE, R. V. COX, J. D. JOHNSTON, and L. A. SELTZER

(Manuscript received April 30, 1982)

A digital speech coder has been designed for real-time operation at a data rate of 9.6 kb/s. The design is based on a combination of two speech compression techniques: Time-Domain Harmonic Scaling (TDHS) and Sub-Band Coding (SBC). It is a highly modularized hardware implementation using five Bell Laboratories Digital Signal Processor (DSP) integrated circuits as the key processing elements. Three DSPs are used in the encoder for pitch detection, TDHS compression, and sub-band encoding. Another two DSPs are used in the receiver for sub-band decoding and TDHS expansion. In this paper we describe the overall design of the system and discuss some of the techniques used to realize it in the DSP hardware in real time. General issues of the algorithm design, software implementation, and hardware design are considered.

I. INTRODUCTION

The subject of digital speech encoding and bit-rate compression has been one of considerable interest in recent years. Attention has focused strongly on bit rates in the range of 9.6 kb/s for applications where good "communications quality" is required and where robustness across a broad range of background noise conditions and speaker variations is necessary.[1,2] This bit rate appears, at present, to be about the lowest practical rate at which this standard of quality and robustness can be reliably achieved. Below 9.6 kb/s presently known techniques have a noticeable synthetic quality and are considerably more fragile to differences in speakers and background conditions.[1]

Several encoding methods for achieving "communications quality" at 9.6 kb/s have been proposed and studied. Most of these methods involve a considerable amount of signal processing to meet these goals and are thus referred to as "high-complexity" algorithms. Their imple-

mentation in real time typically requires the use of specially designed high-speed digital hardware or array-processing digital computers.[3,4]

In recent studies,[2] we found that a combined technique of sub-band coding (SBC) and time-domain harmonic scaling (TDHS) leads to an encoding approach whose quality is comparable to, or better than, some of the previously studied "high-complexity" algorithms. The interesting aspect of this approach is that it is a combination of two relatively "low-complexity" algorithms that are amenable to real-time implementation using currently available technology. Thus, this TDHS/SBC approach appears to be an attractive, economical candidate for real-time implementation of 9.6-kb/s speech encoding using a relatively small amount of hardware.

In this paper we discuss the design of a 9.6-kb/s speech coder based on the above TDHS/SBC approach. The design is highly modularized around the use of the Bell Laboratories Digital Signal Processor (DSP) integrated circuit.[5,6] Three DSPs are used in the encoder for pitch detection, time-domain harmonic-scaling compression, and sub-band encoding, respectively. Another two DSPs are used in the receiver for synchronization, sub-band decoding, and time-domain harmonic-scaling expansion. Essentially all of the signal processing in the coder is performed by the DSPs with only a minimal amount of support hardware for interfacing, clock generation, and input/output (I/O) buffering.

This paper discusses aspects of the overall design and the design parameters of the coder. Other papers[7-9] discuss in more detail the software implementation of the pitch, TDHS, and SBC algorithms on the DSPs and the architecture of the multiple-DSP hardware used to implement the coder.

II. THE TDHS/SBC ALGORITHM

The TDHS/SBC system is basically a cascade of two different speech-compression algorithms, TDHS and SBC. Figure 1 gives a basic block diagram of this approach. The sampled input signal $s(n)$ is first compressed in bandwidth and sampling rate by the TDHS algorithm to form the intermediate signal $s_c(n)$. This processing is performed in a pitch-synchronous manner. Consequently, a pitch detector is required in the system. The SBC encoder digitally encodes the compressed signal, $s_c(n)$, to form the encoded data. This data, multiplexed with the encoded pitch and appropriate frame-synchronization information, forms a 9.6-kb/s bit-stream which is sent over the digital channel. In the receiver the digital signal is first synchronized into frames and demultiplexed into pitch and SBC data. The SBC data is decoded by the SBC decoder to form the intermediate signal, $\hat{s}_c(n)$, which is a quantized version $s_c(n)$. It is then expanded back to its original bandwidth and

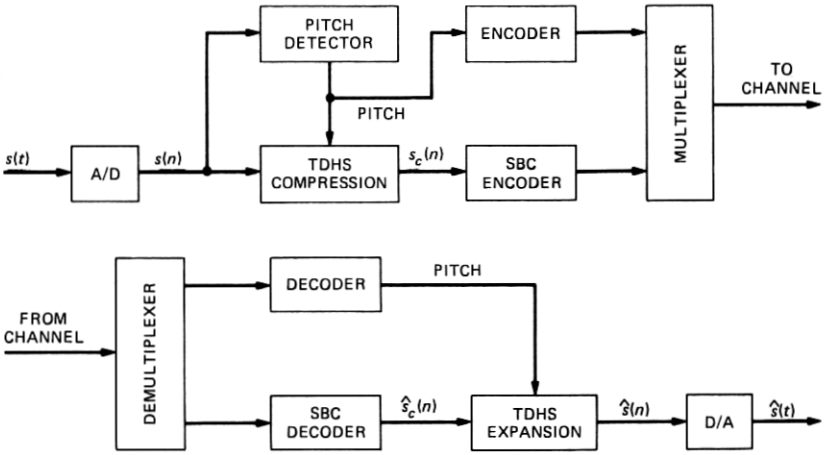

Fig. 1—Basic block diagram of the TDHS/SBC system.

sampling rate by a TDHS expansion algorithm to produce the signal $\hat{s}(n)$, which is a decoded replica of the input $s(n)$.

The two algorithms operate on different properties of redundancy of the speech signal to achieve their compression. The TDHS algorithm takes advantage of the pitch structure or pseudoperiodic nature of speech through a pitch synchronous process.[10] It effectively interpolates two pitch periods of signal into one period to achieve a signal-compression factor of two. Further details of this approach are reviewed in Section V.

The SBC algorithm is a waveform coding technique that achieves a bit-rate compression by adaptively quantizing the speech in frequency bands. It takes advantage of the properties of temporal nonstationarity, spectral-formant structure, and auditory masking in speech production and perception.[1,9,11] Further details of this method are outlined in Section VI.

Because the TDHS and SBC algorithms operate on different properties of redundancy of speech, they are highly complementary and "noncompetitive" in their operation. Thus, the overall compression of the cascaded system is effectively the product of the two individual compression factors.

A second advantage of this cascaded approach is that the degradations introduced by the two compression techniques are perceptually different. Degradations introduced by TDHS compression appear as a form of reverberance, whereas degradations introduced by SBC coding appear in the form of quantizing noise and intermodulation distortion. Since these degradations combine in different perceptual "dimensions," the overall perceived degradation tends to be less objec-

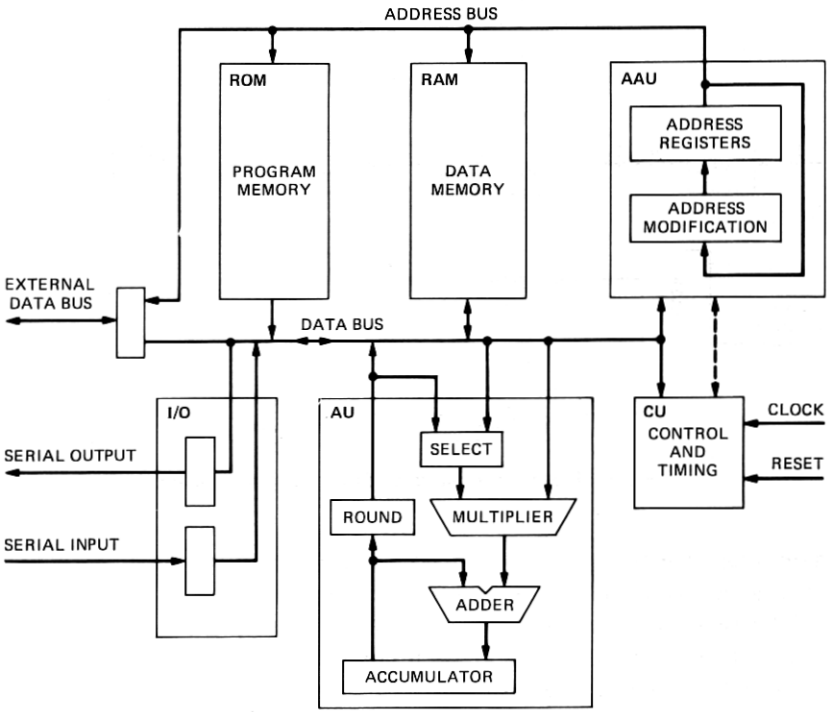

Fig. 2—Block diagram of the DSP.

tionable than if they were combined along the same perceptual "dimension."[2]

Finally, a third desirable feature of this algorithm is that the compression introduced by the TDHS algorithm allows the SBC algorithm to be computed at effectively one-half the computation rate of that which would be required with an uncompressed signal. Thus, it leads to a system that is efficient computationally, as well as one that can be modularized into a system of smaller algorithmic units.

III. BASIC HARDWARE CONFIGURATIONS

The hardware design is highly modularized using the DSP as the key processing element in each module. Therefore, it is useful to review the basic characteristics of this device. Figure 2 shows a basic block diagram of the DSP.[5] The main elements are: (*i*) a 1024-word × 16-bit read-only memory (ROM) for instruction and coefficient storage, (*ii*) a 128-word × 20-bit random-access memory (RAM) for variable data storage, (*iii*) an address arithmetic unit (AAU) with address registers for controlling memory access, (*iv*) a data arithmetic unit (AU) with

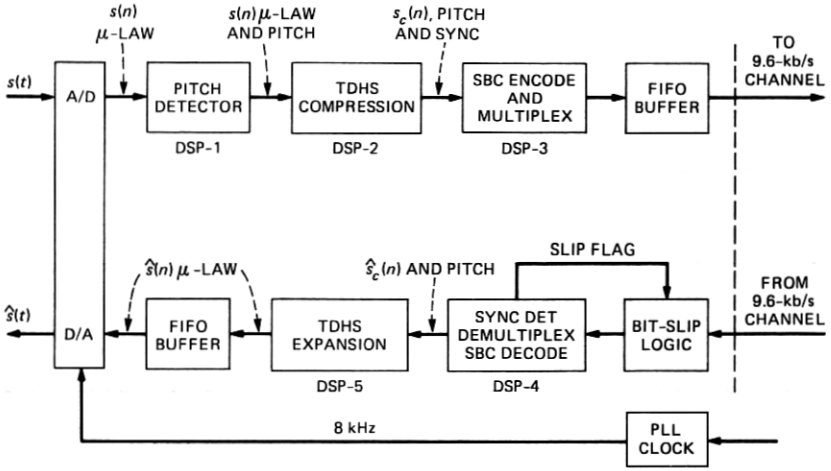

Fig. 3—Block diagram of the hardware configuration for the 9.6-kb/s coder.

provision for multiplication, full product accumulation, rounding, and overflow protection, (*v*) an I/O unit to control serial data transmission in and out of the circuit, and (*vi*) a control unit that provides instruction decoding and processor synchronization.

The processor operates with an 800-ns machine cycle time, which is established by a 5-MHz clock. In one machine cycle it can: (*i*) decode an instruction, (*ii*) fetch data and perform a 16 × 20-bit multiplication, (*iii*) accumulate the output products from the multiplier, and (*iv*) store data in memory.

Figure 3 is a simplified block diagram of the hardware architecture for the 9.6-kb/s coder. It consists of cascaded connections of DSP modules with data passing from one DSP to the next in multiplexed form. The encoder contains three DSPs in series with an analog-to-digital (A/D) converter at the input and a first-in-first-out (FIFO) buffer at the output to the channel. The decoder contains two DSPs with a FIFO buffer between the second DSP and the output to the digital-to-analog (D/A) converter. In addition, a logic circuit is necessary at the receiver between the input of the 9.6-kb/s channel and the first DSP for use in bit slipping for synchronization on startup.

The analog input, $s(t)$, is first converted to digital form $s(n)$ by a μ-law A/D converter.[12] An 8-KHz clock signal for the sampling rate of the A/D converter is generated from the 9.6-KHz channel clock using a phase-locked loop (PLL) circuit. The first DSP is used to implement the pitch detector. It passes the μ-law signal $s(n)$ through to the second DSP along with multiplexed pitch information. The second DSP is used for the TDHS compression algorithm. Its output consists of the com-

pressed signal, $s_c(n)$, multiplexed with pitch and synchronization information. The third DSP is used for the SBC encoder and for multiplexing the pitch, synchronization, and SBC-encoded data into a 9.6-kb/s serial bit stream. Its output is passed to the channel in the form of 16-bit serial words. A FIFO memory is used to control the flow of data between the DSP and the channel.

In the decoder the fourth DSP receives the serial bit stream directly from the 9.6-kb/s channel in the form of 16-bit words. A bit-slipping logic circuit is used to interface the channel to the DSP; it is controlled by a flag from the DSP. The circuit is used to align the first bit of a frame of data to the first bit of a 16-bit word in the synchronization locking mode of the decoder. Once frame synchronization is established, the fourth DSP is used to demultiplex the pitch and SBC data and perform the computation for the SBC decoder. The SBC decoded output, $\hat{s}_c(n)$, and the pitch are then passed to the fifth DSP, which computes the TDHS expansion algorithm. The output, $\hat{s}(n)$, of this DSP is the decoded version of $s(n)$ and it is passed to the D/A (in μ-law PCM format) through a second FIFO buffer.

In the next five sections we discuss each of the components of this system in more detail. Section IV discusses the design of the pitch detector, Section V discusses the TDHS algorithms, and Section VI discusses the design of the SBC. The overall framing and multiplexing structure for the coder is discussed in Section VII, and Section VIII discusses the synchronization detection algorithm used for frame alignment in the receiver. Finally, Section IX discusses the performance of the system.

IV. THE PITCH DETECTOR

The pitch detector is based on a modification of the autocorrelation-type pitch-detection algorithm,[13] and it is designed for implementation in a single DSP.[7] Figure 4 illustrates this basic approach. The input speech signal $s(n)$ is first converted to a linear PCM signal and lowpass filtered to remove spectral energy above 1 KHz. The resulting low-pass filtered signal, $x(n)$, is then used to compute the autocorrelation-function estimate at time n

$$r_n(m) = \sum_{\ell=-\infty}^{\infty} f(n-\ell)x(\ell)x(\ell-m), \tag{1}$$

where m denotes the autocorrelation lag and $f(n)$ corresponds to the analysis window over which $r_n(m)$ is computed. The pitch period is then defined as the lag m_0 over the range of the allowed set of lag values $\{m\}$ for which $\tilde{r}_n(m)$ is maximum, i.e.,

$$\tilde{r}_n(m_0) = \max_m [r_n(m) \cdot g(m)] \tag{2a}$$

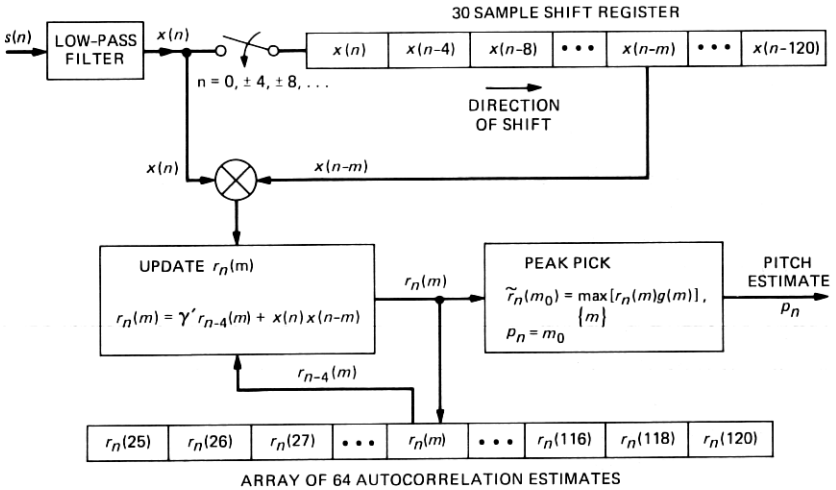

Fig. 4—Block diagram of the pitch-detector implementation.

$$\text{pitch} = p_n = m_0, \tag{2b}$$

where $g(m)$ is a weighting factor used to control the behavior of the algorithm.[7]

A total of 64 autocorrelation coefficients are included in the set $\{r_n(m)\}$ corresponding to values

$$\{m\} = \{25, 26, 27, \cdots 55, 56, 58, 60, 62, \cdots 116, 118, 120\}. \tag{3}$$

This allows the pitch period to be encoded into a 6-bit word for transmission over the channel. With a sampling rate of 8 KHz, the pitch values in the set $\{m\}$ correspond to a pitch frequency range from 66.7 Hz to 320 Hz, which spans the range of most speakers. Note also from eq. (3) that the allowed values of pitch period are more closely spaced (quantized) for low values, m, than for high values. This allows for a more uniform percentage of accuracy of pitch for high- and low-pitch speakers.

The autocorrelation estimate $r_n(m)$ is computed using an exponential window function

$$f(n) = \begin{cases} \gamma^n & n \geq 0 \\ 0 & n < 0. \end{cases} \tag{4}$$

This forms allows $r_n(m)$ to be sequentially updated according to the relation

$$r_n(m) = \gamma r_{n-1}(m) + x(n)x(n-m). \tag{5}$$

In this manner, only two multiplications and one addition are required to update each autocorrelation coefficient.

To further reduce the amount of computation and storage required in the DSP, eq. (5) was modified so that $r_n(m)$ is updated only every fourth sample, i.e.,

$$r_n(m) = \gamma' r_{n-4}(m) + x(n)x(n-m), \tag{6}$$

where γ' typically has a value of $\gamma' = 0.95$. This reduces the total amount of computation by a factor of 4. It also reduces storage requirements for the delayed signal $x(n-m)$ by allowing $x(n)$ to be decimated (reduced in sampling rate) by a factor of 4. Thus, at any one sample time only certain values of $r_n(m)$ can be updated. Computation is therefore distributed over a four-cycle process in which four input samples $s(n)$ are received, all values of $r_n(m)$ are updated, and a new estimate of pitch is determined every four sample times. A more detailed description of this computational structure and the manner in which it is implemented in the DSP is described in Ref. 7. Further details concerning the weighting parameters $g(m)$ and the performance of this design are also available in the same reference.

The speech signal $s(n)$ is passed from the pitch detector to the TDHS algorithm along with the 6-bit encoded pitch information, which is inserted after every fourth speech sample. Since the TDHS algorithm does not require information regarding the voiced or unvoiced nature of the speech, no voiced/unvoiced decision is made by the pitch detector. During voiced regions, the pitch detector measures the speech periodicity, and during the unvoiced and silence regions it gives the best estimate of any long-term correlation that may exist in the signal, even though this correlation may be low.

V. THE TDHS ALGORITHM

The TDHS algorithm compresses the input signal $s(n)$ by a factor of two such that the compressed signal $s_c(n)$ contains one-half of the original number of samples.[2,8,10] Since the sampling rate of $s(n)$ is 8 KHz, the sampling rate of $s_c(n)$ is, therefore, 4 KHz. This compression process can be interpreted in the frequency domain in terms of a 2:1 compression of the spectral bandwidth such that the original 0- to 4-KHz bandwidth of the signal $s(n)$ is scaled to a 0- to 2-KHz bandwidth allowing the sampling rate to be reduced by the factor of two. The compression is achieved by reducing the frequency spacing between pitch harmonics and appropriately scaling the envelope of the spectrum by a factor of two.

In an alternative time-domain interpretation, the signal $s(n)$ is compressed by a factor of two by computing one pitch period of the compressed signal $s_c(n)$ from a weighted average of every two pitch periods of the input signal $s(n)$ in a pitch-synchronous manner. The

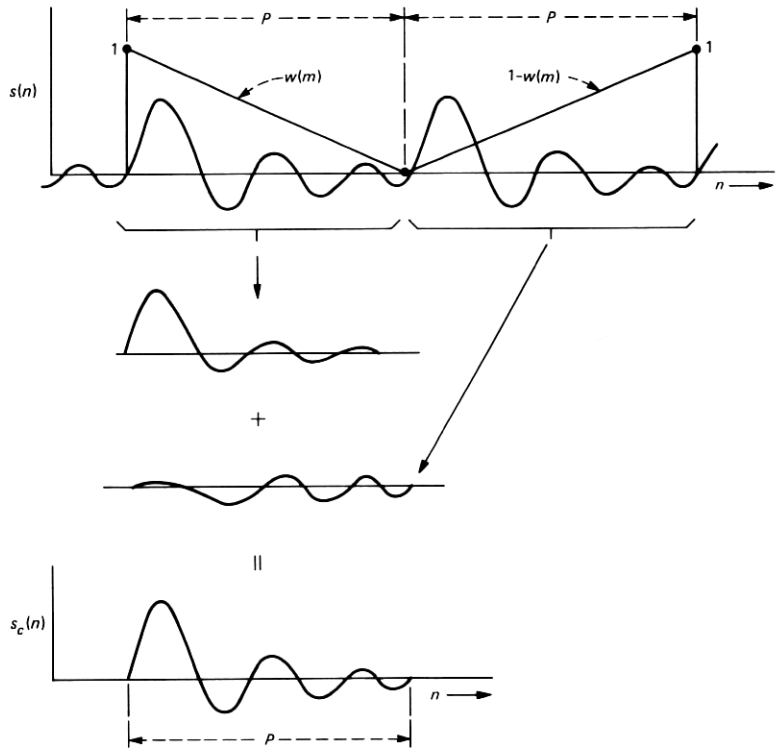

Fig. 5—TDHS compression.

TDHS algorithm is implemented according to this interpretation. Figure 5 illustrates this interpolation process for the compression algorithm. Given the pitch period, $p = p_n$, from the pitch detector, the input speech $s(n)$ is divided into blocks of $2p$ samples. One block of p samples of compressed signal $s_c(n)$ is then computed from these $2p$ samples according to the following process. The first block of p samples of $s(n)$ is weighted by a p-sample window, $w(m)$, $m = 0, 1, \cdots p - 1$, which linearly decreases from a value of 1 to 0 across the block. The second block of p samples of $s(n)$ is similarly weighted with a window $1 - w(m)$ that linearly increases from 0 to 1 across the block. The sum of the two weighted blocks then produces one block of p samples of the compressed signal $s_c(n)$, as illustrated in Fig. 5. The waveform of $s_c(n)$, therefore, looks mostly like the first block of $s(n)$ at its beginning and mostly like the second block of $s(n)$ at the end. In this way the concatenation of the blocks of $s_c(n)$ forms a continuous waveform without end effects from block to block. The next block of $s_c(n)$ is computed in the same manner as above using the next $2p = 2p_n$ samples of $s(n)$.

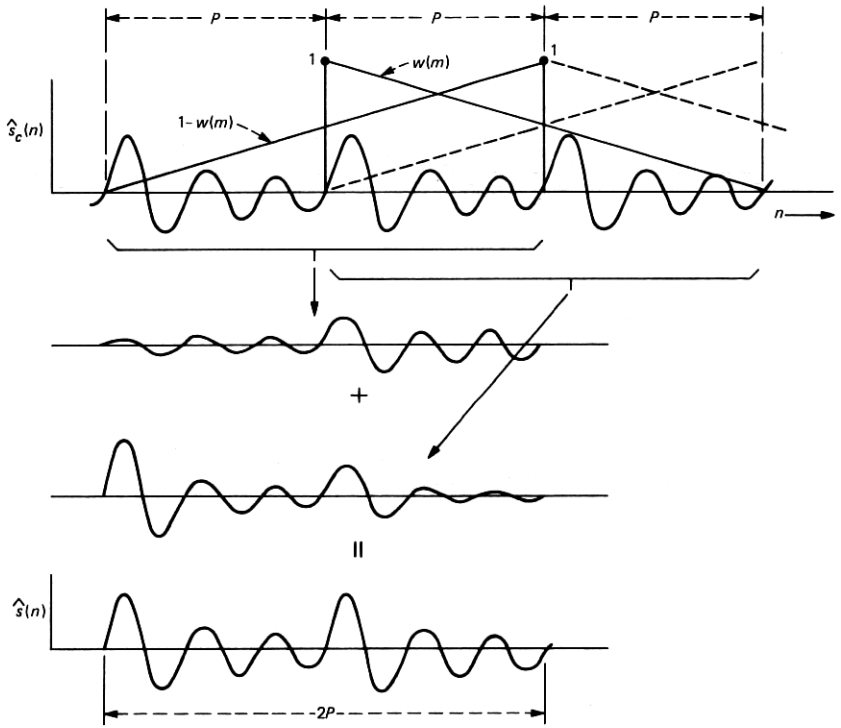

Fig. 6—TDHS expansion.

In the TDHS expansion of $\hat{s}_c(n)$ to $\hat{s}(n)$ a similar pitch-synchronous interpolation is performed. Figure 6 illustrates this process. In this case, $3p$ samples of $\hat{s}_c(n)$ are used to compute $2p$ samples of $\hat{s}(n)$ using the $2p$-sample overlapped windows shown by the solid lines in Fig. 6. The windows are then moved over by p samples, as shown by the dashed lines in Fig. 6, and the next $2p$ samples of $\hat{s}(n)$ are computed in a similar process. Thus, for every p samples of new input signal, $\hat{s}_c(n)$, $2p$ samples of the expanded signal $\hat{s}(n)$ are computed. A careful analysis reveals that this process results in an output waveform $\hat{s}(n)$ that is continuous across the concatenated output blocks without end effects.

Although the TDHS compression and expansion algorithms have been discussed above in terms of block-processing operations, they are more conveniently implemented in the DSP in a stream-processing manner.[8] That is, for every two input samples of $s(n)$ in the TDHS compression, one sample of $s_c(n)$ is computed. Similarly, for every input sample of $\hat{s}_c(n)$ in the TDHS expansion, two outputs of $\hat{s}(n)$ are computed. These operations are performed using the structure shown

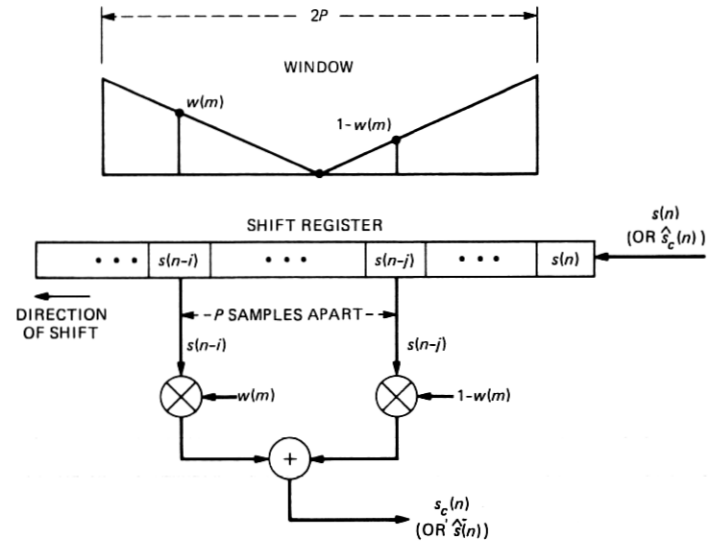

Fig. 7—Block diagram of the implementation of TDHS compression (or expansion).

in Fig. 7. A length $2P$ shift register is used to hold the input data $s(n)$ (for compression) or $\hat{s}_c(n)$ (for expansion), where

$$P = \max p_n = 120. \tag{7}$$

Input samples enter from the right and are shifted one sample to the left for each input. The computation of each output sample involves the multiplication of samples located at indices i and j in the shift register with the respective window coefficients $w(m)$ and $1 - w(m)$, as illustrated in Fig. 7. The sum of these two products is then the interpolated output sample. It also should be clear from the above discussion that the indices i and j are always spaced p samples apart.

For TDHS compression the data in the shift register moves two samples to the left and the indices i and j are decremented (shifted left) by one sample for each output. At the start of a new block of p samples, index i is initialized to the center of the shift register, p is initialized to p_n, j is initialized to $i + p_n$ and $w(m)$ is initialized to $w(0) = 1$. The process is then repeated for the next p samples.

For TDHS expansion the opposite process occurs. Since the output samples are generated at twice the rate of the input samples, the indices i and j must be incremented (shifted right) by 1 after each output sample. At the start of a new block, index j is initialized to the center of the shift register and i is initialized to $j = p$. Further details on how these operations are implemented in the DSP are covered in another paper.[8]

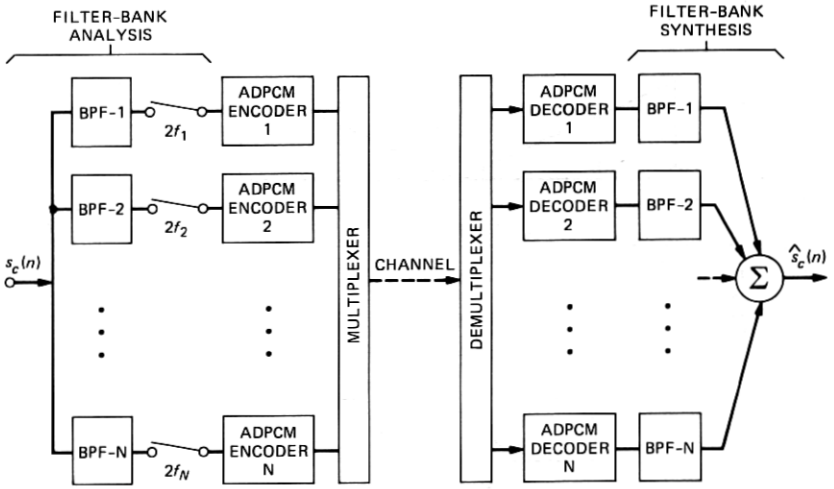

Fig. 8—Block diagram of the SBC encoder and decoder.

Another novel aspect of the above DSP implementation of TDHS compression and expansion algorithms involves the amount of memory required for the shift register. From eq. (7) and Fig. 7 we can see that the required length of the shift register is $2P = 240$ samples, whereas the DSP contains only 128 locations of RAM. This conflict is conveniently solved by using the capability of the DSP to convert between 8-bit μ-law and 20-bit linear PCM formats. By storing the data in the shift register in 8-bit μ-law format and packing two 8-bit words into each 20-bit RAM location, the effective memory of the DSP is doubled, allowing the TDHS algorithms to be completely implemented within individual DSPs without external memory. Also, by carefully organizing the "high" and "low" 8-bit words of memory it is possible in the TDHS algorithms to associate "high" memory with one of the indices i or j and "low" memory with the other. This greatly simplifies the software implementation.[8]

VI. THE SUB-BAND CODER

The sub-band coder (SBC) encodes the compressed signal $s_c(n)$ into a digital bit stream. It is a waveform coding technique that takes advantage of the temporal and spectral properties of speech production and speech perception by partitioning the signal into a set of sub-bands by a filter bank (see Refs. 1, 9, 11, and 14). Each sub-band is effectively bandpass filtered, low-pass translated to *dc*, sampled at its Nyquist rate (twice the width of the sub-band), and then digitally encoded using adaptive differential PCM (ADPCM). Figure 8 illustrates

Table I—Sub-band coder design

Band	Frequency Range (Hz) Uncompressed	Compressed	Sampling Rate (Hz)	Bits/ Sample	Bit Rate
1	0–500	0–250	500	5	2500
2	500–1000	250–500	500	4	2000
3	1000–2000	500–1000	1000	2	2000
4	2000–3000	1000–1500	1000	2	2000
TOTAL	0–3000				8500

a simplified block diagram of this process. In the receiver the digital signal $\hat{s}_c(n)$ is reconstructed by decoding the sub-band signals, interpolating them, translating them back to their original spectral locations, and then summing them to form the decoded signal $\hat{s}_c(n)$.

The sub-band framework offers several advantages. Quantization noise is contained in bands to prevent masking of one frequency band by quantizing noise in another frequency band. Separate adaptive quantizer step sizes are used so that bands with lower signal energy have lower quantizer step sizes and contribute less quantization noise. By appropriately allocating bits in different bands, the shape of the quantization noise can be controlled in frequency. In the lower frequency bands, where pitch and formant structure must be accurately preserved, a larger number of bits/sample are used, whereas in upper frequency bands, where fricative and noise-like sounds occur in speech, fewer bits/sample are used.

A four-band SBC design is used in the 9.6-kb/s coder. Table I summarizes the choice of bands, sampling rates, and bits/sample used in this design. Bandwidths are given with respect to both the uncompressed and compressed frequency scales. Therefore, the total bandwidth of the coder is 3 KHz, relative to the uncompressed frequency scale. The total bit rate of the SBC coder is 8.5 kb/s, leaving the remaining 1.1 kb/s for transmission of pitch and framing information.

Figure 9a illustrates the manner in which the SBC analysis filter bank is implemented. The design is based on the use of quadrature mirror filter (QMF) designs,[9,15] which are implemented in terms of polyphase structures.[16] The QMF approach allows a signal band to be divided into two equally spaced, high-pass filtered (HPF) and low-pass filtered (LPF) sub-bands, which are each reduced in sampling rate by a factor of two. This process is accomplished with a pair of symmetric finite impulse response (FIR) high-pass and low-pass filters. Because of the symmetry and the quadrature mirror relationship of these two filters, their coefficients are identical except for the signs of the odd-numbered coefficients. This property allows the computation to be shared between the two filters by separately computing the even taps, $h_e(n)$, and the odd taps, $h_0(n)$, of the low-pass filter $h(n)$.[9,15] The sum of these two partial computations gives the output for the lower band

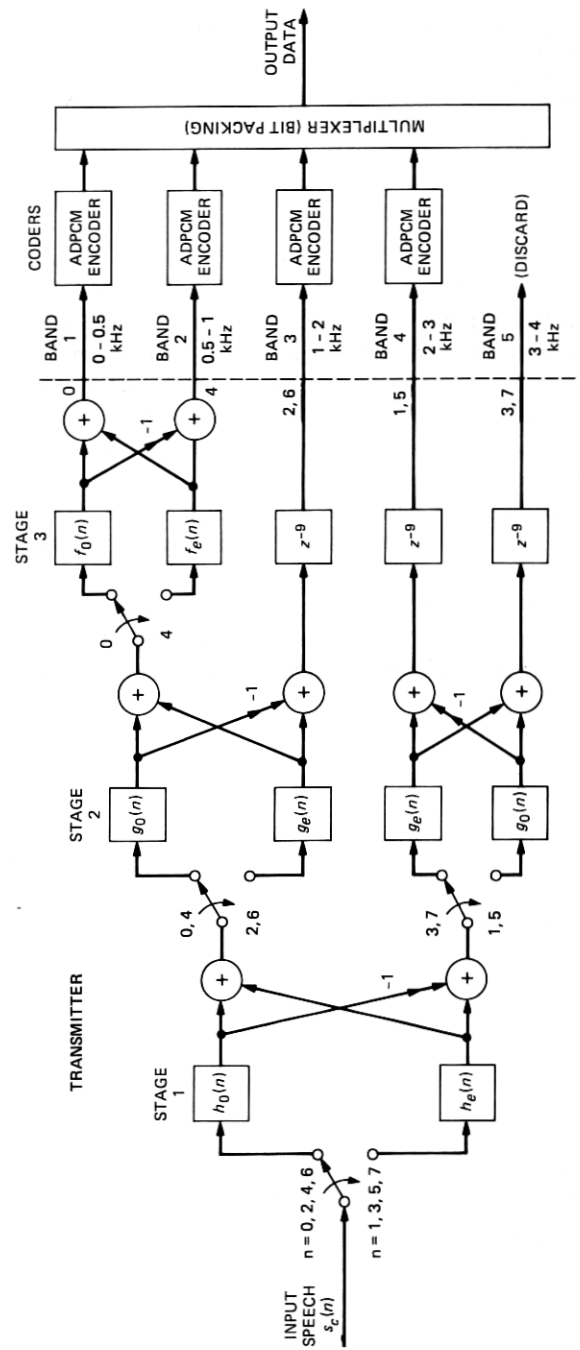

Fig. 9 (a)—QMF filter bank tree structure for analysis.

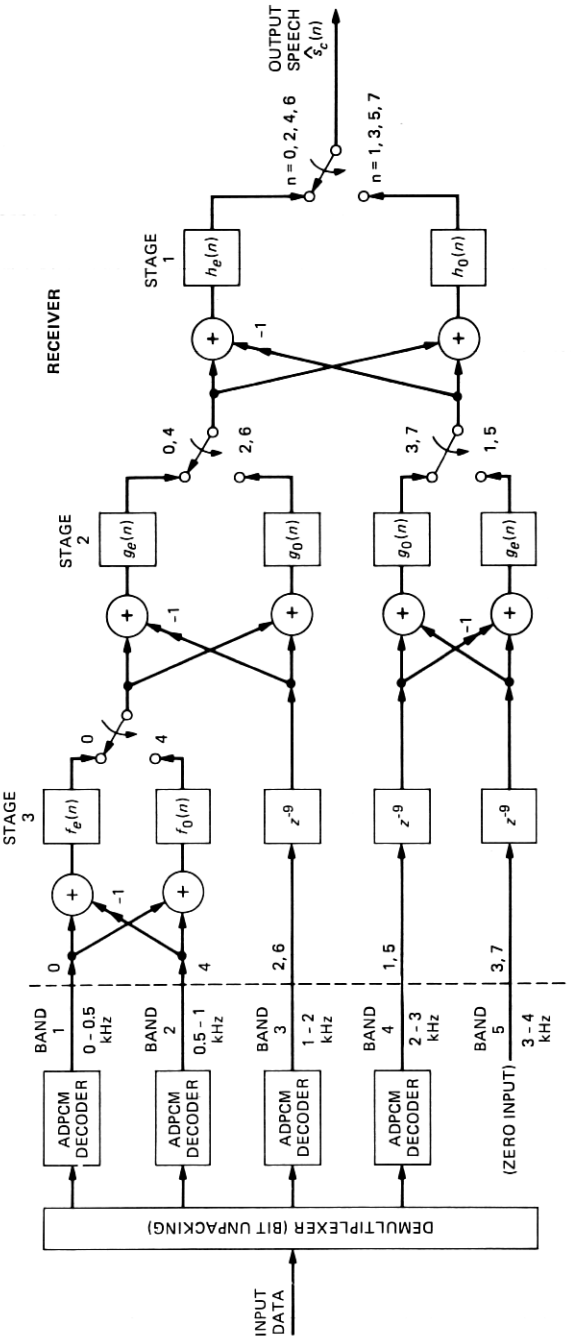

Fig. 9 (b)—QMF filter bank tree structure for synthesis.

Table II—Filter coefficients for $h(n)$, $g(n)$, and $f(n)$

n	$h(n)$	$g(n)$ and $f(n)$
0	0.0006506	0.0065257
1	−0.0013508	−0.0204875
2	−0.0012601	0.0019911
3	0.0041581	0.0464768
4	0.0014272	−0.0262756
5	−0.0093636	−0.0992955
6	−0.0001722	0.1178666
7	0.0178820	0.4721122
8	−0.0041094	
9	−0.0311553	
10	0.0144688	
11	0.0529093	
12	−0.0392449	
13	−0.0998001	
14	0.1284651	
15	0.4664583	

and the difference gives the output for the upper band. By applying this two-band splitting process in the tree structure in Fig. 9a, the band structure in Table I is obtained where the highest band (band 5) is discarded in the process. In the receiver a similar tree structure (Fig. 9b) is used in reverse order to recombine the sub-bands into a full-band signal. The delays in bands 3 and 4 in Fig. 9 are used to compensate for the processing delay caused by the extra split used to generate bands 1 and 2.

The above QMF filter-bank design offers two advantages. First, it leads to an efficient means of computing the filters by sharing computation among bands. Second, because of the quadrature nature of the design, aliasing terms, which are generated in the process of sampling rate reduction in the encoder, are canceled by imaging terms, which are generated in the filter-bank interpolation process in the receiver. This property allows the use of relatively low-order finite impulse response (FIR) filter designs in the coder.

The low-pass filter design, $h(n)$, for the first stage of QMF splitting is accomplished with a 32-tap FIR filter. The coefficients for this design[17] are given in Table II, column 2. Note that only the first symmetric half of the coefficients is given. The remaining coefficients can be obtained from the relation $h(n) = h(31 - n)$ for $n = 16$ to 31. The filter designs $g(n)$ and $f(n)$ for the second and third stages of QMF splitting are accomplished with identical 16-tap FIR filters. Table II, column 3, gives the coefficients for this design. Again, only the first symmetric half of the coefficients is given and coefficients from $n = 8$ to 15 can be obtained from the relation $g(n) = g(15 - n)$. All coefficient values are quantized to the 16-bit accuracy of the DSP in the actual implementation.

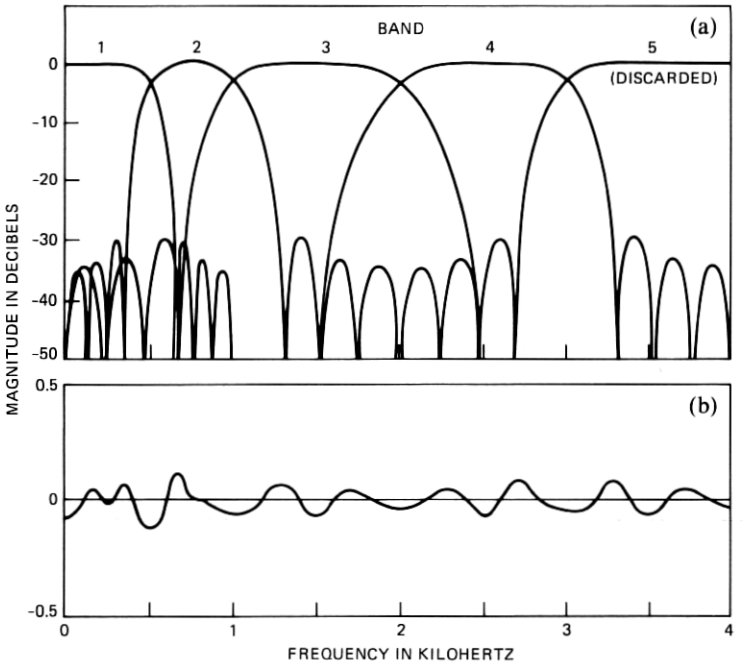

Fig. 10—Frequency response for (*a*) the compositive five-band filter bank, and (*b*) the analysis/synthesis reconstruction based on the filter bank.

The frequency response for the combined filter bank is shown in Fig. 10a, where the band structure given in Table I is clearly apparent. Note also that because the filters in the second and third stages of the structures in Fig. 9 are implemented at lower sampling rates, their frequency responses are respectively scaled to narrower widths in the combined system. Finally, Fig. 10b shows a plot of the total frequency response of the back-to-back filter bank (without ADPCM coding). A total reconstruction error of less than 0.25 dB is observed.

The ADPCM coders for quantizing the sub-band signals are identical to the DSP design discussed in Ref. 18, except for the choice of design parameters. Therefore, we will only briefly review the design and discuss the choice of parameters. Figure 11 is a block diagram of this design. The main elements of the coder are: (*i*) a *b*-bit PCM quantizer (where *b* can be varied from 2 to 5), (*ii*) tables to store the step size and inverse step size, (*iii*) a step-size adaptation circuit to control the choice of step size (i.e., the addresses to the tables), and (*iv*) a first-order fixed predictor.

A predictor signal $p(n)$ is first computed by scaling the previous decoded sub-band signal $\hat{s}_c(n-1)$ by the predictor coefficient β, where for telephone speech, values of β will be close to zero (e.g., 0, −0.4, 0,

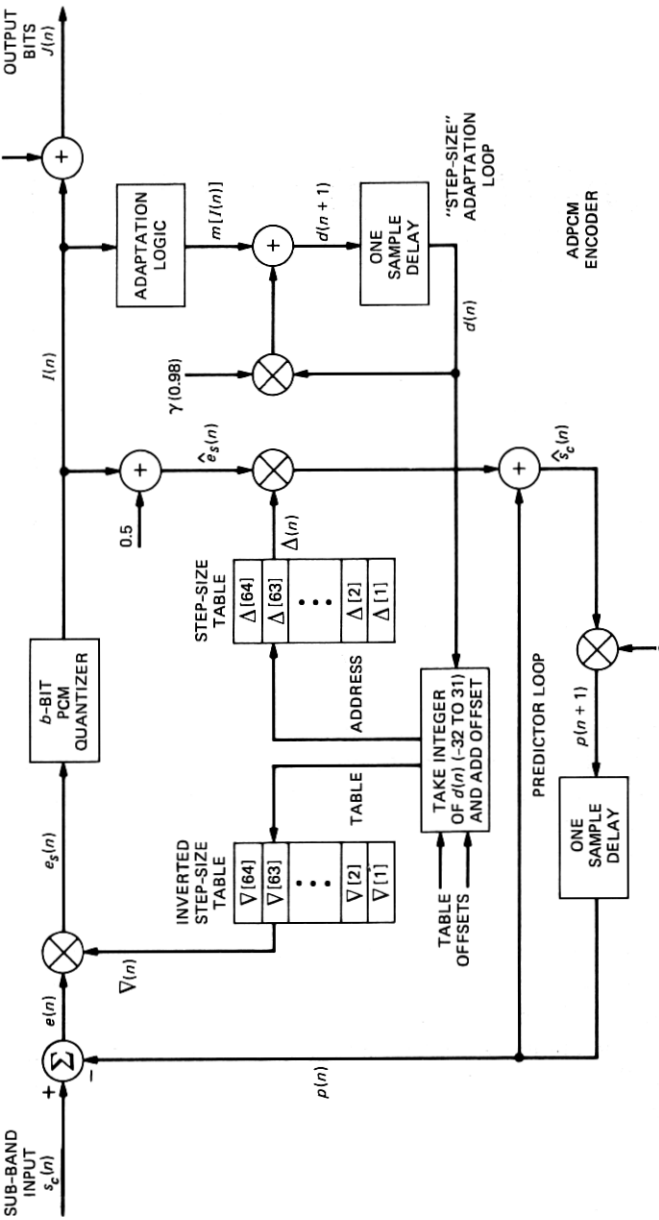

Fig. 11 (a)—Block diagram of the ADPCM encoder.

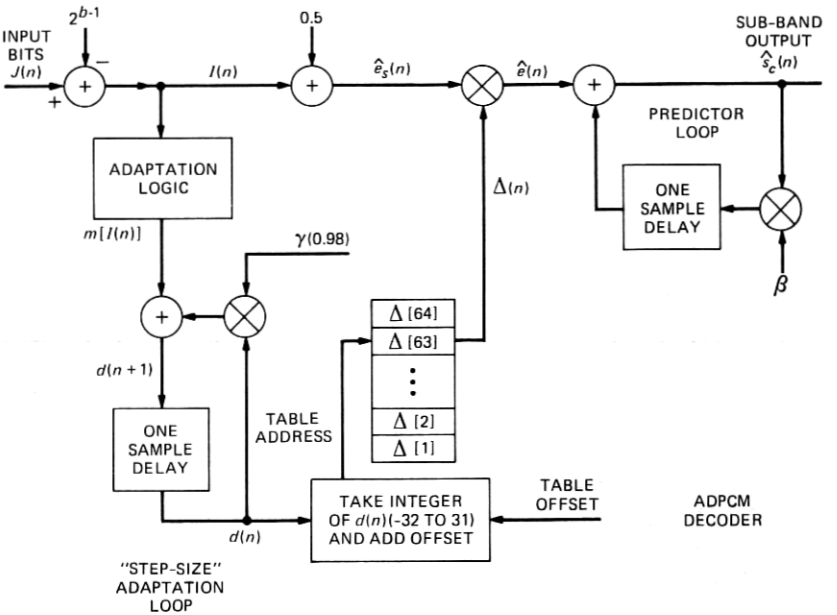

Fig. 11 (b)—Block diagram of the ADPCM decoder.

0 for bands 1 to 4^{11}). The signal $p(n)$ is then subtracted from the sub-band signal $s_c(n)$ to produce the difference $e(n)$, which is adaptively quantized. This is accomplished by first scaling $e(n)$ down by the inverse step size $\nabla(n)$ and limiting and rounding it (PCM coding) to the b-bit integer $I(n)$. By adding the value 2^{b-1} to $I(n)$ a positive b-bit code word, $J(n)$, is obtained, which can be conveniently multiplexed with code words from other sub-bands (as well as pitch and synchronization information).

The b-bit integer, $I(n)$, is also used in the encoder and decoder for obtaining the decoded output $\hat{s}_c(n)$ and for step-size control. Decoding is accomplished by adding 0.5 to $I(n)$ and scaling it back up by the step size $\Delta(n)$. By adding the predictor value $p(n)$ to this quantized difference we obtain the decoded signal $\hat{s}_c(n)$, which is the output of the decoder. It is also used in both the encoder and decoder for generating the next predictor value.

The step size $\Delta(n)$ and its inverse $\nabla(n)$ are determined by the table address, which is adaptively incremented or decremented by the step-size control. The size of the tables are 64 words and they span a 60-dB step-size range (i.e., 0.9375 dB/word). Their address is obtained from the integer part of the variable $d(n)$ (see Fig. 11), which is limited to the range -32 to $+31$. The value of $d(n)$ is stored in a "leaky integrator" with a leak factor of 0.98. In this way $d(n)$ "drifts" toward zero (the

Table III—Quantizer adaptation parameters

No. of Bits	m_1	m_2	Attack Rate (dB/sample)	Decay Rate (dB/sample)
2	4.88	−0.98	4.6	−0.92
3	5.47	−1.27	5.1	−1.2
4	5.47	−1.27	5.1	−1.2
5	5.47	−1.27	5.1	−1.2

center of the tables) and mitigates the effects of channel error. If an outermost (positive or negative) quantizer level is used, a positive value, $m[I(n)] = m_1$ is added to $d(n)$, and if an innermost (positive or negative) level of the quantizer is used, a negative value, $m[I(n)] = m_2$ is added to $d(n)$. In this way the step size is dynamically varied to match the range of $e_s(n)$ to the range of the PCM quantizer. Table III shows typical values of m_1 and m_2 for $b = 2$ to $b = 5$ bit quantizers in SBC and their corresponding "attack" and "decay" rates (in dB/sample) at which they can respectively expand or contract their step sizes.

The manner in which the computation in the SBC is performed is strongly determined by the three-stage framework of the filter bank. Since the sampling rate is reduced by a factor of two at each stage, computation in different stages must be performed at different rates. This is accomplished by distributing the total computation of the coder (and decoder) over an 8-cycle process. In each cycle one input sample $s_c(n)$ is received in the coder, one path of the filter band "tree" is computed, and one ADPCM coder is implemented. The cycle numbers $n = 0, 1, 2, \cdots 7$ in Fig. 9 indicate the paths of computation taken in each cycle. For example, in cycle 0 of the SBC encoder, filters $h_0(n)$, $g_0(n)$, and $f_0(n)$ in the uppermost branch of the tree are updated and the ADPCM coder in band 1 is computed. As can be seen from this structure, stages and coders in the tree that have higher sampling rates are computed more often and those with lower sampling rates are computed less often.

The tree structure also determines the manner in which output bits from the ADPCM coders are multiplexed and framed for transmission. Table IV lists the cycle number and the sub-band that is coded in each cycle. After eight cycles of processing, eight input samples are received and 17 output bits are generated. The process is then repeated.

VII. MULTIPLEXING AND FRAMING OF THE DATA

The 9.6-kb/s data from the TDHS/SBC coder is multiplexed into frames of 96 bits each for serial transmission over the channel. Each frame corresponds to 10 ms of encoded speech. The multiplexing is performed in the third DSP along with the multiplexing of the sub-band coder data. The output from this DSP is in the form of 16-bit words

Table IV—Order of computation in the sub-band coder

Cycle	Band	Bits
0	1	5
1	4	2
2	3	2
3	(5)	—
4	2	4
5	4	2
6	3	2
7	(5)	—

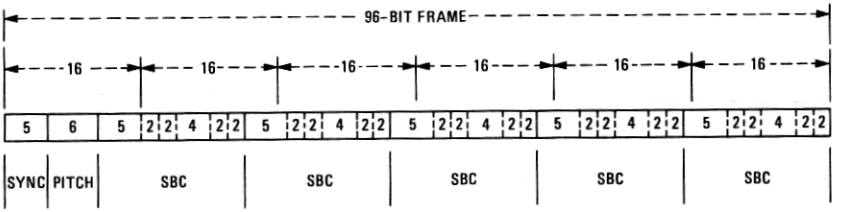

Fig. 12—Frame structure for the 9.6-kb/s bit stream.

such that six words form one 96-bit frame. Figure 12 summarizes this multiplexing structure for one frame of data.

The first five bits in the frame represent a synchronization header, which is used for frame locking in the receiver. The next six bits are used to transmit pitch from the pitch detector. The remaining bits are used for the transmission of five groups of 17 bits each of SBC data, where each group comprises one 8-cycle computational loop in the SBC coder. From Fig. 12 we can see that one 16-bit word can be obtained from the DSP after encoding band 1 in each 8-cycle loop of the SBC coder, and a final 16-bit word can be transmitted at the end of the frame. In the receiver a similar decoding process is performed in the demultiplexer of the SBC decoder.

VIII. SYNCHRONIZATION DETECTION AND DATA ALIGNMENT

Before the data in the receiver can be decoded, the framing structure in the bit stream must first be identified and the data must be aligned with the 16-bit input words according to Fig. 12. This synchronization detection and data alignment is accomplished in the first DSP of the receiver with the aid of the bit-slipping logic at the input from the channel (see Fig. 3).

Figure 13 shows a block diagram of this process. The algorithm is divided into two modes of operation, a synchronization search mode and a run mode. In the search mode the DSP receives input from the channel in the form of 16-bit words. If the first five bits match the bit

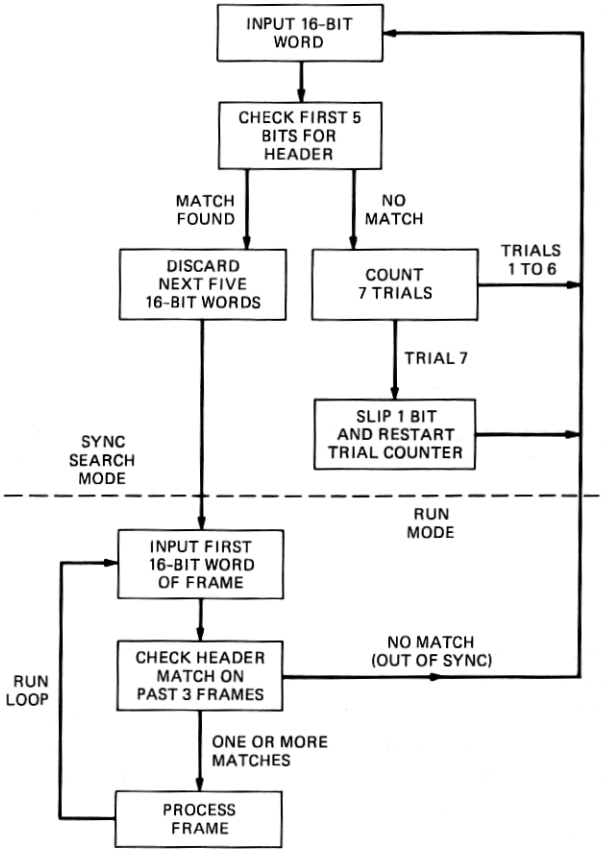

Fig. 13—Block diagram of the synchronization detection and data alignment algorithm.

pattern of the header, it is assumed that the first word of a frame has been found. It then discards the next five words of the frame and proceeds to the run mode.

If no match is found, it waits for the next 16-bit input and tries again. If it is unsuccessful in finding a header after seven trials it assumes that the frame structure is out of alignment with the word structure. It then sends a flag to the bit-slipping logic, which discards one bit in the bit stream. The search process is then repeated. A total of seven trials are used to avoid the possibility of skipping a bit in the middle of the header.

Once the algorithm is in run mode it continues to check the header at the beginning of each frame. If a match is found in one or more of the past three frames it assumes that the algorithm is synchronized

and proceeds to process the frame. In this way the algorithm is more resistant to channel errors that may occur in the header. If no match is found after three consecutive frames, it assumes that the system is out of synchronization (or had received a false start) and it returns to the search mode.

The above algorithm works well and uses a minimum amount of code in the DSP. The bit pattern for the header was chosen to be the 5-bit sequence 10010.

IX. PERFORMANCE

In terms of quality, the real-time DSP version of the TDHS/SBC coder matches the quality of the computer simulations.[2] The degradations introduced by TDHS compression/expansion are generally perceived as a form of reverberance, and degradations introduced by the SBC coder are generally perceived as a form of quantization noise and harmonic distortion.

In a recent study involving TDHS/SBC coding of telephone network speech it was found that the perceived reverberance caused by the TDHS processing was more noticeable than that for high-quality microphone speech.[19] This implies that some caution must be exercised in applications involving a direct tandem connection of TDHS/SBC with the telephone network environment. Later we will show how this effect can be mitigated to some extent. For applications where the characteristics of the transducer can be controlled, this poses no problem.

The reason for the increased reverberance for telephone network speech was found to be a consequence of the strong pre-emphasis caused by the 500-type telephone set specifications and the tight bandpass filtering (200 to 3200) of the D-channel bank. Figure 14 shows the resulting frequency response of the network environment.[19] This can be compared to the flat response of a high-quality microphone.

The network frequency response of Fig. 14 has two effects on the perceived quality of TDHS processing. Both effects stem from the fact that TDHS depends strongly on an accurate pitch measurement for its performance, as well as the assumption that voiced speech can be modeled as a pseudoperiodic signal. For example, if Δf represents the error in the measurement of the fundamental pitch harmonic (caused by measurement error or quantization of the pitch) the nth harmonic will have an error of $n\Delta f$. This means that TDHS processing will always degrade the high-frequency regions of speech more than the low-frequency regions. For high-quality microphone speech this high-frequency reverberance is not very apparent because for voiced sounds the high-frequency regions are relatively low in amplitude. In addition, the strong energy in the first formant helps to perceptually mask any degradations in the high-frequency regions. During unvoiced regions

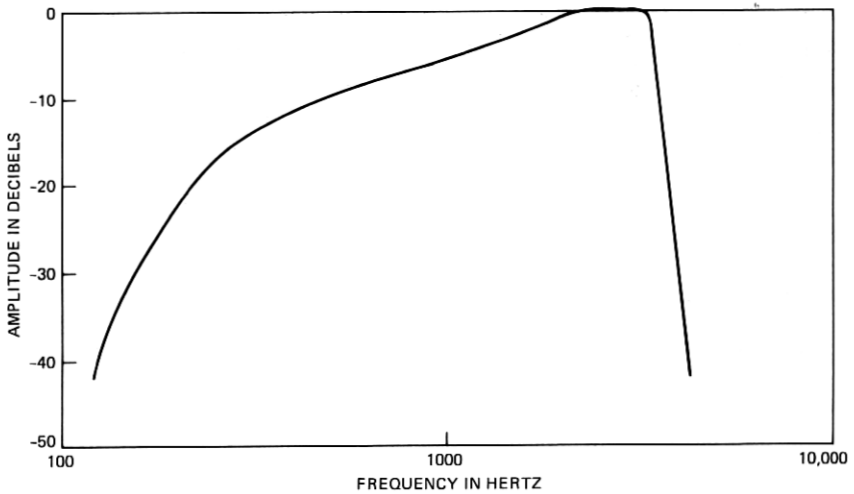

Fig. 14—Frequency response of the telephone network environment.

the noise-like character of speech tends to mask most of the effects of TDHS processing.

The effect of the telephone pre-emphasis in Fig. 14 is to amplify the high-frequency region of the speech by a factor of 15 to 20 dB relative to the low-frequency region. This also amplifies the high-frequency reverberance of TDHS as well. The effect of the high-pass filtering from 0 to 200 Hz effectively removes most of the fundamental harmonic of the speech and consequently reduces the masking process. Thus, both effects contribute to enhancing the perceived reverberance of the TDHS processing. Therefore, the problem is one of auditory perception rather than the TDHS algorithm performing differently on telephone or microphone speech. We found that if high-quality microphone speech is filtered with the same filter response as that of Fig. 14, either before or after TDHS/SBC coding, we get the same effect as with telephone network speech.

The effects of the pre-emphasis in Fig. 14 on TDHS processing can be mitigated to some degree by undoing some of the pre-emphasis. It cannot restore the fundamental harmonic, however, which is removed by 0- to 200-Hz high-pass filtering. Thus, the perceived reverberance can be reduced in amplitude (along with the amplitude of the high-frequency content of the speech) but the masking effects caused by the fundamental harmonic cannot be restored. The deemphasis filter used in the DSP was a simple first-order filter with the difference equation

$$y(n) = 0.65y(n-1) + 0.667x(n) + 0.25x(n-1). \quad (8)$$

This filter can be inserted or removed in the DSP realization by controlling one of the input flags to the DSP with a switch.

A final observation that we have made is that the TDHS/SBC coder appears to perform slightly better with electret microphone speech than with carbon-button microphone speech. We speculate that this is due to the increased harmonic distortion of the carbon-button microphone over that of the electret.

X. CONCLUSIONS

In this paper we have discussed the overall aspects of the design of the 9.6-kb/s TDHS/SBC speech coder. An attractive feature of the design is that it can be broken down into a set of five highly modularized parts, each of which is implemented in a single DSP and effectively fully utilizes the capabilities of the DSP. It therefore leads to a relatively efficient and low-complexity approach to a 9.6-kb/s coder. It offers good "communications" quality that is competitive with that of other higher complexity systems.

The overall processing delay of the coder and decoder is about 80 ms or about 160 ms for a full-duplex system. This amount of delay is fairly typical compared with other competitive techniques at this bit rate.

REFERENCES

1. J. L. Flanagan, M. R. Schroeder, B. S. Atal, R. E. Crochiere, N. S. Jayant, and J. M. Tribolet, "Speech Coding," IEEE Trans. on Commun., *COM-27,* No. 4 (April 1979), pp. 710–37.
2. D. Malah, R. E. Crochiere, and R. V. Cox, "Performance of Transform and Sub-Band Coding Systems Combined with Harmonic Scaling of Speech," IEEE Trans., Acoustics, Speech, and Signal Processing, *ASSP-20,* No. 2 (April 1981), pp. 273–83.
3. R. V. Cox and R. E. Crochiere, "Real-Time Simulation of Adaptive Transform Coding," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-29,* No. 2 (April 1981), pp. 147–54.
4. J. J. Wolf and K. D. Field, "Real-Time Speech Coder Implementation at 9.6 and 16 KB/S," Proc. 1981 IEEE Int. Conf. Acoust., Speech, and Signal Processing (April 1981), pp. 607–10.
5. J. R. Boddie et al., "Digital Signal Processor: Architecture and Performance," B.S.T.J., *60,* No. 7 (September 1981), pp. 1449–62.
6. Special Issue on the DSP, B.S.T.J., *60,* No. 7 (September 1981).
7. R. V. Cox and R. E. Crochiere, "A Single Chip Speech Periodicity Detector," Proc. IEEE 1982 Int. Conf. on Acoustics, Speech, and Signal Processing (May 1982), pp. 525–8.
8. R. V. Cox, R. E. Crochiere, and J. D. Johnston, "An Implementation of Time Domain Harmonic Scaling with Application to Speech Coding," Proc. IEEE 1982 Int. Conf. on Communications (June 1982), pp. 4G.1.1–4.
9. R. E. Crochiere, "Digital Signal Processor: Sub-Band Coding," B.S.T.J., *60,* No. 7 (September 1981), pp. 1633–53. See also: R. E. Crochiere, M. Randolph, J. W. Upton, and J. D. Johnston, "Real-Time Implementation of Sub-Band Coding on a Programmable Integrated Circuit," Proc. IEEE 1981 Int. Conf. on Acoustics, Speech, and Signal Processing, (March 1981), pp. 455–8.
10. D. Malah, "Time-Domain Algorithms for Harmonic Bandwidth Reduction and Time Scaling of Speech Signals," IEEE Trans. Acoust., Speech, and Signal Processing *ASSP-27* (April 1979), pp. 121–33.

11. R. E. Crochiere, S. A. Weber, and J. L. Flanagan, "Digital Coding of Speech in Sub-Bands," B.S.T.J., *55,* No. 8 (October 1976), pp. 1069–85.
12. D. G. Marsh, B. K. Ahuja, T. Misawa, M. R. Dwarakanath, P. E. Fleischer, and V. R. Saari, "A Single-Chip CMOS PCM Coder with Filters," IEEE J. of Solid State Circuits, *SC-16,* No. 4 (August 1981), pp. 308–15.
13. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals,* Englewood Cliffs, NJ: Prentice-Hall, 1978.
14. R. E. Crochiere, R. V. Cox, J. D. Johnston, "Real-Time Speech Coding," IEEE Trans. on Commun., *COM-30,* No. 4 (April 1982), pp 621–34.
15. D. Esteban and C. Galand, "Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes," Proc. 1977 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (May 1977), pp. 191–5.
16. M. G. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital Filtering by Polyphase Network: Application to Sample Rate Alteration of Filter Banks," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-24,* No. 2 (April 1976), pp. 109–114.
17. J. D. Johnston, "A Filter Family Designed for Use in Quadrature Mirror Filter Banks," Proc. IEEE Int. Conf. ASSP (April 1980), pp. 291–4.
18. J. R. Boddie, J. D. Johnston, C. A. McGonegal, J. W. Upton, R. E. Crochiere, and J. L. Flanagan, "Digital Signal Processor: "Adaptive Differential Pulse-Code-Modulation Coding," B.S.T.J., *60,* No. 7, Part 2 (September 1981), pp. 1547–61.
19. W. R. Daumer, "Subjective Evaluation of Several Efficient Speech Coders," IEEE Trans. on Commun., *COM-30,* No. 4 (April 1982), pp. 655–62.